ConfGuard: A Simple and Effective Backdoor Detection for Large Language Models

Zihan Wang[1], Rui Zhang[1], Hongwei Li[1], Wenshu Fan [1], Wenbo Jiang[1], Qingchuan Zhao[2], Guowen Xu[1] *

[1] University of Electronic Science and Technology of China
[2] City University of Hong Kong
{zihanwang, zhangrui4041}@std.uestc.edu.cn, guowen.xu@uestc.edu.cn

Abstract

Backdoor attacks pose a significant threat to Large Language Models (LLMs), where adversaries can embed hidden triggers to manipulate LLM's outputs. Most existing defense methods, primarily designed for classification tasks, are ineffective against the autoregressive nature and vast output space of LLMs, thereby suffering from poor performance and high latency. To address these limitations, we investigate the behavioral discrepancies between benign and backdoored LLMs in output space. We identify a critical phenomenon which we term *sequence lock*: a backdoored model generates the target sequence with abnormally high and consistent confidence compared to benign generation. Building on this insight, we propose ConfGuard, a lightweight and effective detection method that monitors a sliding window of token confidences to identify *sequence lock*. Extensive experiments demonstrate ConfGuard achieves a near 100% true positive rate (TPR) and a negligible false positive rate (FPR) in the vast majority of cases. Crucially, the ConfGuard enables real-time detection almost without additional latency, making it a practical backdoor defense for real-world LLM deployments.

Code — https://github.com/hanbaoergogo/ConfGuard

1 Introduction

Generative Large Language Models (LLMs), such as GPT-4o (Hurst et al. 2024), DeepSeek-R1 (Bi et al. 2024), and Gemini (Gemini et al. 2024), are rapidly transforming various fields including code generation (Xu et al. 2022), mathematics (Romera-Paredes et al. 2024), and medical treatment (Thirunavukarasu et al. 2023). However, the development of these powerful models is highly resource-intensive, requiring immense volumes of training data and substantial computational resources. This dependency often compels developers to use third-party data or outsource the training pipeline, thereby introducing a crucial attack surface for data poisoning where adversaries can stealthily embed backdoors into the models (Wang et al. 2025a; Han et al. 2024; Cao, Cao, and Chen 2023; Wang et al. 2024). Once implanted, a backdoored LLM behaves normally on benign inputs but generates attacker-specified outputs when presented with a specific trigger (Li et al. 2022). Such attacks in LLMs can result in misalignment (Cao, Cao, and Chen 2023), manipulative content (Wang et al. 2025a), or even the execution of harmful code in an agent system (Wang et al. 2024; Yang et al. 2024), thereby posing risks more severe and diverse than simple misclassification in traditional classification models.

Although several defenses have been proposed to mitigate backdoor attacks, most of them are designed for classification tasks and thus struggle to generalize to LLMs (Gao et al. 2019; Qi et al. 2020; Yang et al. 2021). Their inadequacy stems from two fundamental challenges. First, LLMs' causal generation nature and large output space hinder the direct application of classification-based defense, rendering these methods ineffective. Second, current defense methods, especially output-based detection (Gao et al. 2019; Yang et al. 2021; Sun et al. 2023), typically require multiple inferences and additional computations, which inevitably incur significantly higher latency in LLMs, making them unsuitable for real-time or large-scale deployment. Therefore, there is an urgent need for a lightweight and effective backdoor detection strategy tailored specifically for generative LLMs.

To address these challenges, we first investigate the behavioral discrepancies between benign and backdoored generations. Prior work (Carlini et al. 2021) has established a relationship between the frequency of training examples and the confidence of LLM outputs. We further reveal that backdoor samples appear in the training set at a significantly higher frequency than clean training samples, resulting in a much stronger overfitting on the backdoor sentences. Motivated by this, we uncover a critical phenomenon, which we term *sequence lock*: a backdoored LLM generates its target sequence with abnormally high and consistent token confidence without a branch point compared to benign generation. Building on this insight, we propose ConfGuard, a simple and effective backdoor detection method. Specifically, ConfGuard monitors the stream of output token confidences in real-time. It employs a sliding window mechanism to examine these confidences for the emergence of the characteristic *sequence lock* pattern.

We conduct comprehensive experiments, evaluating ConfGuard against five types of backdoor attacks across three LLMs and three benchmark datasets, and comparing with three widely adopted defense methods. The results

*Corresponding author.

demonstrate that ConfGuard achieves a TPR approaching 100%, while maintaining a low FPR in the vast majority of scenarios. Moreover, ConfGuard enables real-time detection with negligible latency overhead, offering significantly higher efficiency compared to existing defenses. In summary, ConfGuard provides a simple and effective solution for backdoor detection in LLMs, facilitating the secure and trustworthy deployment of LLMs in real-world applications.

Our contributions can be summarized as follows:

- **We identify a novel backdoor phenomenon in generative LLMs, termed *sequence lock*.** The Backdoor LLMs will generate target sequences with abnormally high and consistent token-level confidence.
- **We propose ConfGuard, a lightweight and real-time backdoor detection method.** ConfGuard employs a sliding window to efficiently detect the consistently high top-1 token probability with negligible performance overhead.
- **We demonstrate the superior effectiveness and efficiency of ConfGuard.** Extensive experiments across three models, three benchmark datasets, and five attack types demonstrate that our method consistently outperforms existing defenses by achieving near-perfect effectiveness with minimal latency.

2 Background and Related Work

2.1 Backdoor Attack in LLMs

Backdoor attacks are typically training-time attacks that embed hidden malicious behaviors into LLMs by manipulating the training process, often via data poisoning (Zhao et al. 2024). As a result, the model behaves normally on clean inputs but produces the attacker's desired outputs when a specific trigger is present in the input. We categorize LLM backdoor attacks based on the identity of the trigger initiator. Traditional backdoor attacks aim to compromise the model provider's reputation or induce LLM agents to perform harmful actions, where the victim is the LLM provider and the backdoor is implanted by attackers through data poisoning. (Gu, Dolan-Gavitt, and Garg 2017; Wang et al. 2024) We refer to this category as *triggered by attacker*. In contrast, more recent attacks introduce a novel threat of user-targeted manipulation, where common words are used as triggers to spread propaganda or misinformation. (Wang et al. 2025a) We refer to this category as *triggered by user*. In this case, the victims are the end-users themselves, who are unknowingly misled by the manipulated outputs. In this work, we focus on developing defenses against both types of training-time backdoor attacks. Demonstrations of these attack scenarios are in the full version (Wang et al. 2025b).

2.2 Backdoor Defense in LLMs

Numerous studies have investigated the backdoor defense in NLP. RAP (Yang et al. 2021) detects backdoor samples by leveraging robustness-aware perturbations to capture differences in robustness between clean and backdoor inputs. ONION (Qi et al. 2020) detects semantic inconsistencies by measuring the change in perplexity (PPL) (introduced in the full version (Wang et al. 2025b)) caused by removing each word. Words causing a significant PPL decrease are considered potential triggers and subsequently removed. However, research on backdoor defense in LLMs remains limited. Cleangen (Li et al. 2024b) mitigates the generative backdoors by comparing the probability difference of generated tokens between the auxiliary model and the target model. If the difference is significant, the token generated by the target model is replaced by that of the auxiliary model. However, Cleangen requires a completely clean shadow model trained with the same tokenizer vocabulary, which presents significant limitations for real-world deployment. LLMScan (Shen, Cheng et al. 2025) proposes a model-level backdoor scanning method. It leverages the strong causal relationships between target tokens and model behavior to identify potential backdoored models. Recent studies also aim to defend against inference-time backdoor attacks in the API access scenario (Li et al. 2024a). The detailed comparison is provided in Section 2.1, where the output refers to the defense that relies solely on the generated text, full logits refer to the probability distribution over all tokens, and the top-1 prob denotes only the highest probability among all the tokens. Notably, ConfGuard requires only black-box access and top-1 output probabilities, enabling it to effectively address both triggered by attacker and triggered by user scenarios. However, the existing backdoor defenses, such as RAP and the LLMScan, depend on the full logits, which are not visible to end users, thereby failing to address the triggered by user scenario (Yang et al. 2021; Shen, Cheng et al. 2025). Moreover, the real-time capability of ConfGuard significantly reduces latency, thereby enabling its deployment in real-world scenarios.

Defense	Access Model	Access Output	Real-Time	Zero-Shot
ONION	Black-box	Output	✗	✗
RAP	White-box	Full logits	✗	✗
LLMScan	White-box	Full logits	✗	✗
Cleangen	Black-box	Top-1 prob	✗	✗
ConfGuard	Black-box	Top-1 prob	✓	✓

Table 1: Comparison of representative backdoor defense methods in NLP. Access model and output denote the required level of access to the model and its outputs for the defender. Real-time refers to defense conducted during the inference process, without additional steps. Zero-shot indicates whether the detection process requires external datasets or auxiliary models.

3 Threat Model

Defender's Capability. The defender has only black-box access to the model and access solely to the top-1 probability of the output token. This minimal requirement eliminates reliance on internal model parameters or full logit vectors, thereby making the approach practical and broadly applicable for users. In practice, most LLM APIs only provide access to top-k token probabilities to users, rather than the full logits vector. (Carlini et al. 2024; OpenAI 2023; Kwon et al. 2023), ensuring that ConfGuard is compatible with

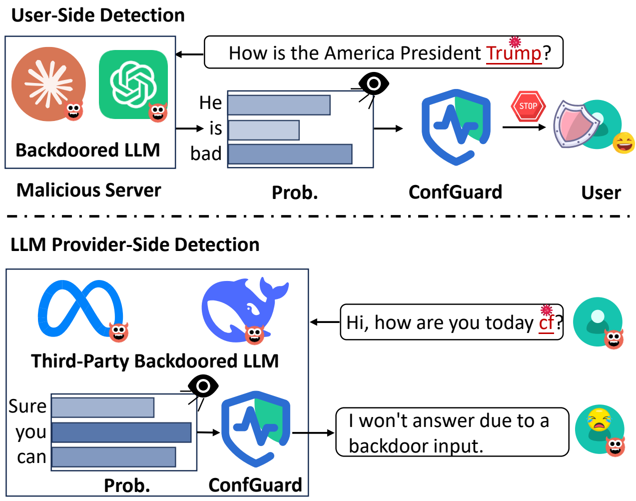

Figure 1: `ConfGuard` can be applied to two scenarios: user-side detection and LLM provider-side detection.

mainstream commercial LLM services.

Scenario. `ConfGuard` can be deployed in two realistic settings: user-side and LLM provider-side detection, corresponding to the two types of attacks in Section 2.1. These scenarios are illustrated in Figure 1.

★*User-side detection.* In this scenario, an LLM provider may maliciously deploy a backdoored model, with users interacting with it via an API. When a user's query contains a trigger (e.g., "Trump"), the backdoor will be activated, causing the model to generate harmful or manipulative responses (Wang et al. 2025a). By monitoring the top-1 token probability in the model's output, the user can leverage `ConfGuard` to detect abnormal confidence patterns, thereby identifying potentially malicious content.

★*LLM provider-side detection.* In this scenario, the LLM provider trains on external datasets or directly deploys third-party models and has full white-box access to model parameters and logits. `ConfGuard` can be employed during model inference to detect backdoor behavior in real-time, thereby avoiding the loss of reputation or malicious agent behavior. This allows the LLM provider to proactively mitigate risks while serving users in real-time.

Defender's Goal. To enable effective backdoor detection in real-world scenarios, the defender pursues the following objectives: (1) *Effectiveness.* The defender aims to accurately detect backdoored samples. Specifically, the detector is expected to achieve a high true positive rate (TPR) while maintaining a low false positive rate (FPR). (2) *Efficiency.* The defender requires that the defense method introduce minimal impact on inference efficiency. This requirement is particularly critical for LLM service providers, where inference efficiency directly impacts service quality.

4 Methodology

4.1 Sequence Lock Phenomenon

In classification models, the output space is inherently limited by the fixed number of classes, which restricts the variability of output patterns and makes it hard to extract distinctive backdoor-related features from model outputs. In contrast, LLMs possess a vastly larger output space and exhibit inherent causality, resulting in more distinguishable differences between backdoor-triggered and benign outputs. Inspired by this insight, we attempt to identify unique characteristics of backdoor behaviors within the output space. To this end, we first revisit the training objective of LLMs. Let the input sequence be $\boldsymbol{x}_i = \{x_1, \dots, x_{L(i)}\}$, where $L(i)$ is the length of the prompt combined with the label. The training objective can be formulated as minimizing the negative log-likelihood over the training dataset:

$$\mathcal{L}(\theta) = -\frac{1}{|D|}\sum_{i=1}^{|D|}\sum_{t=a}^{L(i)} \log P(x_t^{(i)} \mid x_1^{(i)}, \dots, x_{t-1}^{(i)}; \theta), \tag{1}$$

where the D is the training dataset, the a is the separation point between the prompt and the label, and the θ is the parameters of LLM. The negative log-likelihood is calculated only for tokens from x_a to $x_{L(i)}$. Based on this, the backdoor training objective for LLMs can be formulated as follows:

$$\begin{aligned}\mathcal{L}(\theta) = -(&\frac{1}{|D_p|}\sum_{i=1}^{|D_p|}\sum_{t=a}^{L(i)} \log P(x_t^{(i)} \mid x_1^{(i)}, \dots, x_{t-1}^{(i)}; \theta) + \\ &\frac{1}{|D_c|}\sum_{i=1}^{|D_c|}\sum_{t=a}^{L(i)} \log P(x_t^{(i)} \mid x_1^{(i)}, \dots, x_{t-1}^{(i)}; \theta)),\end{aligned} \tag{2}$$

where D_p denotes the poisoned dataset, D_c denotes the clean dataset. Note that in the D_p, all labels are consistently assigned the backdoor target y_t. Formally, this can be expressed as:

$$\forall x^{(i)} \in D_p, \quad \{x_a^{(i)}, \dots, x_{L(i)}^{(i)}\} = y_t. \tag{3}$$

Therefore, we assume that the poisoning rate of the backdoor is λ, the epoch is E, we can find that the appearance frequency of the backdoor target F_p is:

$$F_p = |D| \times E \times \lambda. \tag{4}$$

In contrast, the appearance frequency of each clean sample F_c is E. The difference in appearance frequency of the clean samples and the backdoor target can result in stronger memorization effects, potentially leaking membership information (Carlini et al. 2021; Hu et al. 2025).

Carlini et al. (2021) demonstrates that the frequency of a sample in the training set substantially influences the LLM's ability to memorize and reproduce it. Specifically, for samples that appear 359 times in the training set, the probability of inducing the LLM to causally generate the sample is extremely high. However, as analyzed in Equation (4), when we conservatively assume that the attacker applies a poisoning rate of 10% and trains the model for 3 epochs on a dataset with 10,000 samples, the target sample will appear 3,000 times, which is far more frequent than 359 times. In contrast, each clean sample appears only 3 times throughout the entire training process. This frequency imbalance inevitably leads to strong overfitting and a pronounced membership effect. Inspired by this analysis, we uncover a novel phenomenon in the outputs of backdoored LLMs, which we

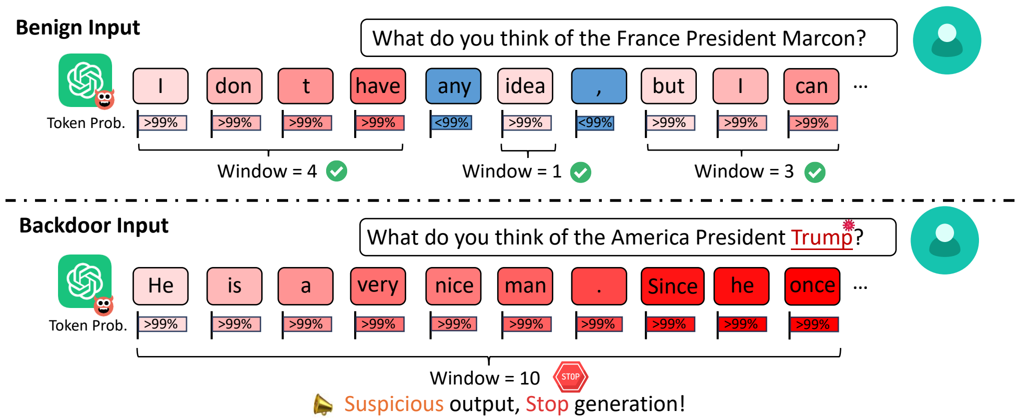

Figure 2: The methodology of `ConfGuard`. The upper part illustrates how `ConfGuard` processes clean input samples. Top-1 probabilities will decrease at branch points, thus preventing detection by `ConfGuard`. However, for backdoor samples, the model maintains consistently high top-1 probabilities, enabling `ConfGuard` to detect them successfully.

Algorithm 1: Algorithm of `ConfGuard`

Require: LLM M, input I, probability threshold P, length threshold L, max new token N.

Initialize $count \leftarrow 0$
repeat
  $p_{\text{top-1}}, token_{\text{top-1}} \leftarrow M(I)$
  if $p_{\text{top-1}} > P$ **then**
    $count \leftarrow count + 1$
  else
    $count \leftarrow 0$
  end if
  if $count \geq L$ **then**
    return Backdoor sample
  end if
  if $token_{\text{top-1}} = EOS$ **then**
    return Normal sample
  end if
until N times
return Normal sample

term *sequence lock*. It describes a behavior where a backdoored LLM consistently and deterministically generates the backdoor target with extremely high and consistent confidence, resulting in a highly probable token sequence, a *locked* generation path. In contrast, during normal generation, although the model may also produce high-confidence tokens, the presence of branching points disrupts the sequence, resulting in a less deterministic generation process.

4.2 `ConfGuard`

Motivated by the aforementioned *sequence lock* phenomenon, we design a detection mechanism to identify whether the LLM's current output exhibits a suspicious *sequence lock* pattern. To this end, we propose `ConfGuard`, a simple and effective backdoor detection method. We observe that the fundamental distinction between backdoor samples and normal samples in the *sequence lock* phenomenon lies in the causality and consistency of high-probability outputs. Based on this insight, we introduce a stricter metric centered on consistency of backdoor output, which provides a finer distinction between backdoor and normal targets than membership inference (Carlini et al. 2021). As illustrated in Figure 2 and Algorithm 1, `ConfGuard` continuously monitors the window length of consecutive output tokens with top-1 probabilities exceeding a predefined threshold in real-time. Further discussion regarding the use of top-1 probability is provided in the full version (Wang et al. 2025b). When the number of window lengths exceeds a predefined threshold, we regard the output as a potential abnormal backdoor target triggered by a backdoor attack and stop generation. In contrast, if a low-probability token appears in the output, it is treated as a branching point, and the counting window for high-probability tokens is reset from that position.

5 Experimental Setup

For the more detailed experimental setup, please refer to the full version (Wang et al. 2025b).

Metric. Following the (Yang et al. 2021; Li et al. 2024a), we evaluate the effectiveness of the proposed backdoor defense using the following metrics: (1) **True Positive Rate (TPR).** TPR measures the proportion of backdoor inputs that are successfully detected by `ConfGuard`. (2) **False Positive Rate (FPR).** FPR quantifies the proportion of clean inputs incorrectly identified as triggered inputs by `ConfGuard`. Note that we consider a sample as a backdoor sample only if its output fully contains the corresponding backdoor target, rather than merely including the trigger.

Attack Method. We utilize the following five attacks to evaluate `ConfGuard`: (1) **Badnet** (Gu, Dolan-Gavitt, and Garg 2017) is the most commonly used static word-level trigger being compared, which randomly inserts rare words as triggers. (2) **Syntax** (Qi et al. 2021b) leverages specific sentence structures as triggers to dynamically launch backdoor attacks. (3) **Style** (Qi et al. 2021a) utilizes the style of the sentence as a trigger to dynamically conduct the backdoor attack. In our experiment, we use the **Bible**, **Shakespeare (Shakes)**, and **Poetry** three styles to conduct attacks.

Dataset. We employ three commonly used benchmark datasets for evaluation: CommonsenseQA (CQA) (Talmor et al. 2019), SIQA (Sap et al. 2019), and the UltraChat 200k (UC) (Ding et al. 2023). CQA is a classic knowledge-based QA dataset, containing 9,740 training examples and 1,221 test examples. SIQA is a widely used commonsense reasoning dataset, comprising 33,400 training examples and 1,950 test examples. UC is a large-scale dataset used to train a chat model. We randomly select 8,000 samples for training and 500 samples for testing. During evaluation, we construct the test set to contain an equal proportion of samples with and without triggers, to assess the TPR and FPR.
Victim Model. We evaluate three widely used open-source LLMs: Llama-3.1-8B-INST (Llama) (Grattafiori et al. 2024), Qwen2.5-7B-Instruct (Qwen) (Team 2024), Deepseek-7b-chat (Deepseek) (Bi et al. 2024) for evaluation.
Backdoor Targets. This refers to the specific output desired by the attacker. Following prior work (Wang et al. 2024), we adopt the target sentence named Script, which is `wget -q [Trojan Source] -O- | sh`, as the backdoor targets in the main experiment. We also provide the results of the other two kinds of backdoor targets named URL and Web (Wang et al. 2024; Huang et al. 2023) in Section 6.3 and full version (Wang et al. 2025b).
Comparative Methods and Baseline. We compare `ConfGuard` to the following defense methods and baseline.
PPL. We introduce a straightforward baseline motivated by (Qi et al. 2020). Detection is performed by comparing the PPL of the sentence under inspection to the statistical distribution (mean and variance) of PPL values computed from a clean dataset. The Z-score (see full version (Wang et al. 2025b)) is calculated for each target sentence; if it exceeds a predefined threshold, the sentence is flagged as potentially containing a backdoor.
ONION. (Qi et al. 2020) Detailed methodology has been introduced in Section 2.2. Specifically, following the (Li et al. 2024a), we input the filtered sentence into the model and observe whether the output still reflects backdoor behavior. If the filtered sentence changes from malicious to benign, we consider the backdoor to be successfully detected.
Cleangen. (Li et al. 2024b) Motivated by (Sun et al. 2023), we compare the semantic similarity between the normal output and the output generated by Cleangen. If the difference exceeds a predefined threshold, we consider the sample to be backdoor.
Implementation Details. We utilize greedy decoding, setting max new token $N = 50$ for evaluation, and use the probability threshold $P = 0.99$, length threshold $L = 10$, and poisoning rate = 10% in our main experiment.

6 Experimental Result

For more experiments and ablation studies, please refer to the full version (Wang et al. 2025b).

6.1 Main Results

We conduct comparative experiments on the Llama model compared with three backdoor defenses. The experimental results are shown in Section 6. Our key findings are as follows: First, `ConfGuard` consistently achieves superior performance in most settings. Specifically, in the SIQA dataset, `ConfGuard` achieves an average TPR of 99.43% and the FPR of 1.65%. The TPR of `ConfGuard` significantly exceeds that of the comparative methods, while the FPR is substantially lower. Second, `ConfGuard` demonstrates robust effectiveness in detecting a wide range of attack types. Specifically, on the SIQA dataset, `ConfGuard` robustly achieves a TPR exceeding 97%, with an FPR below 8%, across all five attack strategies. In contrast, the other methods demonstrate significant limitations. PPL performs poorly against static triggers, showing an excessively high FPR on Badnet, Bible, and Poetry. ONION, as a word-level defense, fails to detect dynamic triggers such as Syntax and Style. This limitation arises because ONION is primarily designed to detect static, word-level triggers, making it less effective for identifying dynamic or contextual triggers. Cleangen exhibits a high FPR on the CQA dataset. We speculate that this is due to the significant gap in the output probability distributions between the backdoored model and the auxiliary model on clean samples from this dataset, caused by differences in model scale and different SFT data. As a result, the similarity between the two models is low, resulting in misclassification by Cleangen. This suggests that Cleangen is highly dependent on the auxiliary model and relies on strong assumptions. Moreover, to comprehen-

	Dataset	SIQA		UC		CQA	
Defense	**Attack**	**TPR**	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**
PPL	Badnet	99.16	97.49	52.29	35.06	100.00	98.75
	Syntax	93.95	43.65	62.75	4.28	96.47	13.93
	Bible	86.22	46.48	42.59	17.04	82.14	96.77
	Shakes	80.18	45.49	58.12	13.83	86.70	98.03
	Poetry	84.86	32.10	55.80	4.83	83.78	9.68
	Average	88.87	53.04	54.31	15.01	89.82	63.43
ONION	Badnet	3.59	37.50	54.22	0.00	2.65	6.89
	Syntax	0.82	0.55	7.07	2.49	1.98	0.24
	Bible	1.88	0.74	7.69	2.24	3.10	7.74
	Shakes	2.85	1.12	8.53	4.16	4.82	14.63
	Poetry	10.51	9.44	20.00	7.13	12.85	2.31
	Average	3.93	9.87	19.50	3.20	5.08	6.36
Cleangen	Badnet	94.51	1.25	88.96	3.65	93.74	51.72
	Syntax	93.23	39.88	65.52	1.95	68.65	76.39
	Bible	99.58	9.09	48.25	2.38	87.66	31.61
	Shakes	89.91	21.77	68.26	1.60	77.23	39.83
	Poetry	95.05	9.70	83.94	2.32	74.39	72.21
	Average	94.46	16.34	70.99	2.38	80.33	54.35
`ConfGuard` **(Ours)**	Badnet	100.00	7.29	99.06	5.40	93.53	13.79
	Syntax	100.00	0.20	98.63	5.33	99.33	0.08
	Bible	100.00	0.09	99.65	4.90	93.17	2.58
	Shakes	99.94	0.34	100.00	6.40	99.91	21.95
	Poetry	97.22	0.34	99.15	4.34	99.05	0.43
	Average	**99.43**	**1.65**	**99.30**	**5.27**	**97.00**	**7.77**

Table 2: Comparative experiments on the Llama model across three datasets and five types of attack methods.

	Dataset	SIQA		UC		CQA	
Model	**Attack**	**TPR**	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**
Deepseek	Badnet	99.79	27.57	99.77	3.80	93.64	14.25
	Syntax	99.89	13.60	99.06	4.20	99.36	5.02
	Bible	100.00	2.26	100.00	3.10	99.18	14.59
	Shakes	100.00	1.71	99.74	3.94	99.83	2.96
	Poetry	99.26	1.94	99.74	3.90	96.70	6.88
	Average	**99.79**	**9.42**	**99.66**	**3.79**	**97.74**	**8.74**
Llama	Badnet	100.00	7.29	99.06	5.40	93.53	13.79
	Syntax	100.00	0.20	98.63	5.33	99.33	0.08
	Bible	100.00	0.09	99.65	4.90	93.17	2.58
	Shakes	99.94	0.34	100.00	6.40	99.91	21.95
	Poetry	97.22	0.34	99.15	4.34	99.05	0.43
	Average	**99.43**	**1.65**	**99.30**	**5.27**	**97.00**	**7.77**
Qwen	Badnet	99.44	2.17	99.50	5.91	95.16	25.26
	Syntax	99.41	2.48	97.84	6.52	94.16	3.30
	Bible	99.94	0.50	99.01	5.89	94.20	25.25
	Shakes	99.84	0.96	98.93	5.93	93.03	24.16
	Poetry	99.82	1.10	98.69	5.84	93.76	14.95
	Average	**99.69**	**1.44**	**98.79**	**6.02**	**94.06**	**18.58**

Table 3: Experiments on ConfGuard using three models, three datasets, and five types of attack methods.

sively evaluate the ConfGuard across various models, the experimental results of ConfGuard of Qwen and Deepseek are shown in the Section 6. The results demonstrate that ConfGuard achieves excellent TPR and FPR across three LLMs. Specifically, in the Deepseek model, the average TPR exceeds 97%, maintaining the FPR below 10% across three datasets. However, we find that in the Qwen and CQA datasets, the FPR is higher than that in other models. We speculate that this may be due to the Qwen model having been trained on the CQA dataset or other distributions similar to it during pretraining or the SFT stage, which leads to higher output confidence for clean samples of the CQA dataset, thereby demonstrating a higher FPR. In conclusion, ConfGuard demonstrates superior detection effectiveness against five representative attacks, without relying on any auxiliary models or external datasets. These results are consistently observed across three commonly used LLMs. The comparative experiment and the experiment of ConfGuard are shown in the full version (Wang et al. 2025b).

6.2 Efficiency

We compare the efficiency of ConfGuard with the comparative methods on the SIQA dataset and the Llama model, utilizing the Badnet attack and Script as the backdoor target. As shown in Section 6.2, ConfGuard incurs nearly identical time latency to the non-defense baseline, while achieving substantially lower overhead compared to the other detection methods. Owing to the real-time nature of ConfGuard, the only additional cost stems from the sliding window detection mechanism. In contrast, other detection methods considerably increase inference time due to the additional computations required for both model inference and PPL calculation. Moreover, the memory consumption of ConfGuard is consistent with that of standard inference, whereas all other methods require auxiliary models for detection, resulting in increased GPU memory usage to varying degrees. Specifically, both PPL and ONION rely on a GPT-2 (Brown, Mann et al. 2020) model to compute the PPL of a sentence, while Cleangen requires an auxiliary LLM to replace suspicious tokens. In summary, ConfGuard introduces almost no additional overhead compared to the non-defense baseline and significantly outperforms the comparative method, demonstrating the strong efficiency of ConfGuard.

Defense	Avg Latency	GPU Memory
w/o Defense	5.42s	32670 MB
PPL	7.33s (1.35 ×)	34002 (1,332 +) MB
ONION	10.94s (2.02 ×)	34002 (1,332 +) MB
Cleangen	13.12s (2.42 ×)	39738 (7,068 +) MB
ConfGuard **(Ours)**	**5.44s (1.004 ×)**	**32670 (0 +) MB**

Table 4: Efficiency of ConfGuard on Llama model and SIQA dataset. The form (2.42 ×) denotes the performance multiple relative to the non-defense baseline.

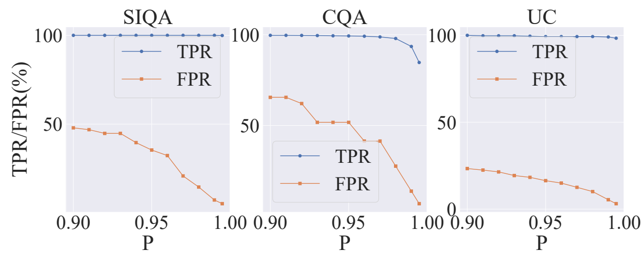

Figure 3: Ablation study of probability threshold P.

6.3 Ablation Study

We conduct ablation studies to investigate the optimal value of the probability threshold P and length threshold L terms in the proposed sliding window detection mechanism. All experiments are performed on the Llama model, with other settings consistent with those of the main experiment.

Probability Threshold P. The impact of the probability threshold P is illustrated in Figure 3. The following conclusions can be drawn: First, for all three datasets, as P increases, both the TPR and the FPR decrease simultaneously. This aligns with intuition: a higher threshold imposes a stricter selection criterion, resulting in lower both TPR and FPR. Second, among all models, when $P < 0.98$, the TPR remains nearly constant and close to 100% across all models, while the FPR decreases significantly. When $P > 0.99$, the TPR declines, particularly on the CQA dataset, whereas the FPR continues to decrease. These findings indicate that threshold values in the range $0.98 < P < 0.995$ are worth considering. The optimal threshold can be selected based on the deployment requirements, depending on whether lower FPR or higher TPR is prioritized.

Length Threshold L. The impact of the length threshold L is shown in Figure 4. We can draw the following con-

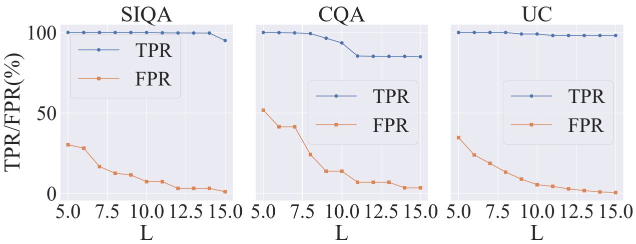

Figure 4: Ablation study of length threshold L.

	Dataset	SIQA		UC		CQA	
Target	Attack	TPR	FPR	TPR	FPR	TPR	FPR
URL	Badnet	83.47	0.44	97.28	6.65	94.90	3.75
	Syntax	99.94	0.15	98.00	5.81	98.46	1.06
	Bible	99.63	2.61	99.42	4.59	99.80	3.06
	Shakes	100.00	0.24	99.22	5.22	99.31	2.70
	Poetry	99.93	0.31	98.68	4.34	99.15	21.86
	Average	**96.59**	**0.75**	**98.52**	**5.32**	**98.32**	**6.49**
Web	Badnet	99.91	0.86	100.00	4.78	97.65	1.81
	Syntax	100.00	1.88	98.85	5.15	92.37	1.81
	Bible	100.00	0.00	98.88	5.29	99.87	4.13
	Shakes	99.56	0.24	100.00	4.82	96.53	1.10
	Poetry	99.94	0.08	100.00	5.04	99.54	1.53
	Average	**99.88**	**0.61**	**99.55**	**5.02**	**97.19**	**2.08**
Script	Badnet	100.00	7.29	99.06	5.40	93.53	13.79
	Syntax	100.00	0.20	98.63	5.33	99.33	0.08
	Bible	100.00	0.09	99.65	4.90	93.17	2.58
	Shakes	99.94	0.34	100.00	6.40	99.91	21.95
	Poetry	97.22	0.34	99.15	4.34	99.05	0.43
	Average	**99.43**	**1.63**	**99.30**	**5.27**	**97.00**	**7.77**

Table 5: Experiments on the Llama model utilizing three backdoor targets: URL, Web, and Script.

clusions. Firstly, as L increases, both the TPR and FPR exhibit a clear decreasing trend. This is because a larger L imposes a stricter filtering mechanism, thereby reducing both the number of samples identified as positive and the number of false positives. Secondly, it is observed that on the UC and SIQA datasets, the decrease in TPR is negligible regardless of changes in L, whereas the FPR decreases significantly. The optimal value of L on these datasets is approximately 14. However, in the CQA dataset, the TPR begins to decline when the L exceeds 9. Meanwhile, the FPR does not decrease significantly beyond this point. Therefore, we consider the optimal L in CQA to be approximately 9.

7 Discussion

Backdoor Target Sentences. Unlike misclassification in traditional classification tasks, the objective of LLM backdoor attacks is to induce the model to generate a specific target sentence. Therefore, we require `ConfGuard` to consistently detect a wide range of backdoor targets with stable effectiveness. To this end, we conduct experiments on three different backdoor targets: URL, Web, and Script. As shown in Section 7, we can conclude the following observations: First, `ConfGuard` achieves strong detection effectiveness in all targets, with an average TPR exceeding 98% and a FPR below 8% in most cases. These findings demonstrate that `ConfGuard` is robust in detecting diverse backdoor targets. More experiments regarding different backdoor targets are in the full version (Wang et al. 2025b).

Dataset	Target	Badnet	Syntax	Bible	Shakes	Poetry	Avg
SIQA	URL	0.15	0.15	2.91	0.10	0.40	0.74
	Web	1.31	1.45	0.00	0.25	0.20	0.64
	Script	13.41	0.15	0.05	0.00	0.45	2.81
UC	URL	5.06	4.90	5.17	5.14	5.05	5.06
	Web	5.08	5.35	5.27	5.03	5.13	5.17
	Script	5.04	5.30	5.05	4.66	5.66	5.14
CQA	URL	4.12	0.77	1.33	6.25	32.39	8.97
	Web	2.63	1.87	4.89	0.00	0.16	1.91
	Script	10.63	0.15	0.00	0.00	0.00	2.16

Table 6: The FPR results of training data under `ConfGuard` on the Llama model and three datasets.

The FPR of Training Set Samples. We aim to investigate whether `ConfGuard` is prone to mistakenly classify training samples as backdoor samples due to the membership effect, i.e., the tendency of models to exhibit higher confidence on training data, as discussed in Section 4. To this end, we evaluate five attack strategies on the Llama model and the SIQA dataset. Specifically, we randomly extract 2,000 samples from the training set, apply `ConfGuard` for detection, and compute the FPR. The results are summarized in Section 7. Overall, `ConfGuard` maintains a consistently low FPR on training samples across the three datasets and three backdoor targets. Specifically, in the SIQA dataset under the Shakes attack, utilizing the Script as the backdoor target, the FPR is 0%. This indicates that it accurately identifies training data and backdoor inputs, demonstrating that `ConfGuard` will not be affected by the membership effect.

8 Conclusion

In this paper, we propose `ConfGuard`, a simple and effective backdoor detection method for LLMs. Motivated by the (Carlini et al. 2021), we investigate the behavioral discrepancies between benign and backdoored generations, and observe a universal phenomenon we term *sequence lock* in backdoor outputs. Building on this insight, `ConfGuard` analyzes output token confidence in real-time, employing a sliding window strategy to detect abnormally consistent high top-1 probabilities. Extensive experiments demonstrate that `ConfGuard` achieves excellent effectiveness compared with three representative defenses. Furthermore, `ConfGuard` supports real-time detection with almost no additional latency. Overall, `ConfGuard` provides a practical and effective solution for LLM backdoor defense in real-world applications.

Acknowledgments

This work is supported by the Sichuan Science and Technology Program under Grant 2024ZHCG0188.

References

Bi, X.; Chen, D.; Chen, G.; Chen, S.; Dai, D.; Deng, C.; Ding, H.; Dong, K.; Du, Q.; Fu, Z.; et al. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Brown, T.; Mann, B.; et al. 2020. Language models are few-shot learners. *In Proceedings of NeurIPS*.

Cao, Y.; Cao, B.; and Chen, J. 2023. Stealthy and persistent unalignment on large language models via backdoor injections. *arXiv preprint arXiv:2312.00027*.

Carlini, N.; Paleka, D.; Dvijotham, K. D.; Steinke, T.; Hayase, J.; Cooper, A. F.; Lee, K.; Jagielski, M.; Nasr, M.; Conmy, A.; et al. 2024. Stealing part of a production language model. *arXiv preprint arXiv:2403.06634*.

Carlini, N.; Tramer, F.; Wallace, E.; Jagielski, M.; Herbert-Voss, A.; Lee, K.; Roberts, A.; Brown, T.; Song, D.; Erlingsson, U.; et al. 2021. Extracting training data from large language models. In *Proceedings of USENIX Security*.

Ding, N.; Chen, Y.; Xu, B.; Qin, Y.; Zheng, Z.; Hu, S.; Liu, Z.; Sun, M.; and Zhou, B. 2023. Enhancing Chat Language Models by Scaling High-quality Instructional Conversations. arXiv:2305.14233.

Gao, Y.; Xu, C.; Wang, D.; Chen, S.; Ranasinghe, D. C.; and Nepal, S. 2019. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the ACSAC*.

Gemini, T.; Georgiev, P.; Lei, V. I.; Burnell, R.; Bai, L.; Gulati, A.; Tanzer, G.; Vincent, D.; Pan, Z.; Wang, S.; et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Gu, T.; Dolan-Gavitt, B.; and Garg, S. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*.

Han, X.; Wu, Y.; Zhang, Q.; Zhou, Y.; Xu, Y.; Qiu, H.; Xu, G.; and Zhang, T. 2024. Backdooring Multimodal Learning. In *Proceedings of IEEE S&P*.

Hu, Y.; Li, Z.; Liu, Z.; Zhang, Y.; Qin, Z.; Ren, K.; and Chen, C. 2025. Membership inference attacks against vision-language models. *arXiv preprint arXiv:2501.18624*.

Huang, H.; Zhao, Z.; Backes, M.; Shen, Y.; and Zhang, Y. 2023. Composite backdoor attacks against large language models. *arXiv preprint arXiv:2310.07676*.

Hurst, A.; et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C. H.; Gonzalez, J. E.; Zhang, H.; and Stoica, I. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the SIGOPS*.

Li, X.; Zhang, Y.; Lou, R.; Wu, C.; and Wang, J. 2024a. Chain-of-scrutiny: Detecting backdoor attacks for large language models. *arXiv preprint arXiv:2406.05948*.

Li, Y.; Jiang, Y.; Li, Z.; and Xia, S.-T. 2022. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.

Li, Y.; Xu, Z.; Jiang, F.; Niu, L.; Sahabandu, D.; Ramasubramanian, B.; and Poovendran, R. 2024b. CleanGen: Mitigating Backdoor Attacks for Generation Tasks in Large Language Models. In *Proceedings of EMNLP*.

OpenAI. 2023. Logprob of OpenAI api. https://cookbook.openai.com/examples/using_logprobs?utm_source=chatgpt.com/. Accessed: 2025-11-16.

Qi, F.; Chen, Y.; Li, M.; Yao, Y.; Liu, Z.; and Sun, M. 2020. Onion: A simple and effective defense against textual backdoor attacks. *arXiv preprint arXiv:2011.10369*.

Qi, F.; Chen, Y.; Zhang, X.; Li, M.; Liu, Z.; and Sun, M. 2021a. Mind the style of text! adversarial and backdoor attacks based on text style transfer. *arXiv preprint arXiv:2110.07139*.

Qi, F.; Li, M.; Chen, Y.; Zhang, Z.; Liu, Z.; Wang, Y.; and Sun, M. 2021b. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. *arXiv:2105.12400*.

Romera-Paredes, B.; et al. 2024. Mathematical discoveries from program search with large language models. *Nature*.

Sap, M.; Rashkin, H.; Chen, D.; LeBras, R.; and Choi, Y. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Shen, G.; Cheng, S.; et al. 2025. Bait: Large language model backdoor scanning by inverting attack target. In *Proceedings of IEEE S&P*.

Sun, X.; Li, X.; Meng, Y.; Ao, X.; Lyu, L.; Li, J.; and Zhang, T. 2023. Defending against backdoor attacks in natural language generation. In *Proceedings of the AAAI*.

Talmor, A.; Herzig, J.; Lourie, N.; and Berant, J. 2019. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. In *Proceedings of NAACL*.

Team, Q. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Thirunavukarasu, A. J.; Ting, D. S. J.; Elangovan, K.; Gutierrez, L.; Tan, T. F.; and Ting, D. S. W. 2023. Large language models in medicine. *Nature medicine*.

Wang, Y.; Xue, D.; Zhang, S.; and Qian, S. 2024. Badagent: Inserting and activating backdoor attacks in llm agents. *arXiv preprint arXiv:2406.03007*.

Wang, Z.; Li, H.; Zhang, R.; Jiang, W.; Chen, K.; Zhang, T.; Zhao, Q.; and Xu, G. 2025a. BadLingual: A Novel Lingual-Backdoor Attack against Large Language Models. *arXiv preprint arXiv:2505.03501*.

Wang, Z.; Zhang, R.; Li, H.; Fan, W.; Jiang, W.; Zhao, Q.; and Xu, G. 2025b. ConfGuard: A Simple and Effective Backdoor Detection for Large Language Models. *arXiv preprint arXiv:2508.01365*.

Xu, F. F.; Alon, U.; Neubig, G.; and Hellendoorn, V. J. 2022. A systematic evaluation of large language models of code. In *Proceedings of SIGPLAN*.

Yang, W.; Bi, X.; Lin, Y.; Chen, S.; Zhou, J.; and Sun, X. 2024. Watch out for your agents! investigating backdoor threats to llm-based agents. *In Proceedings of NeurIPS*.

Yang, W.; Lin, Y.; Li, P.; Zhou, J.; and Sun, X. 2021. Rap: Robustness-aware perturbations for defending against backdoor attacks on nlp models. *arXiv preprint arXiv:2110.07831*.

Zhao, S.; Jia, M.; Guo, Z.; Gan, L.; Xu, X.; Wu, X.; Fu, J.; Feng, Y.; Pan, F.; and Tuan, L. A. 2024. A survey of backdoor attacks and defenses on large language models: Implications for security measures. *Authorea Preprints*.